API 接口说明

（注：Context Caching API 预计 6.30th 正式发布）

Context Caching （上下文缓存）是一种高效的数据管理技术，它允许系统预先存储那些可能会被频繁请求的大量数据或信息。这样，当您再次请求相同信息时，系统可以直接从缓存中快速提供，而无需重新计算或从原始数据源中检索，从而节省时间和资源。

创建 Cache

`POST https://api.moonshot.cn/v1/caching`

请求参数

参数名称	参数类型（以 Python Type Hint 为例）	是否必填	参数说明
model	str	是	模型组（model family）名称。注意，由于被缓存的内容可同时应用于 moonshot 的多个模型（`moonshot-v1-8k`、`moonshot-v1-32k`、`moonshot-v1-128k`），因此这里不指定某个具体的模型，而是指定模型的组名称；当前支持的值为 `moonshot-v1`。

参数名称	参数类型（以 Python Type Hint 为例）	是否必填	参数说明
messages	List[Dict[str, any]]	是	缓存的消息内容，其格式与 `/v1/chat/completions` 接口中的 messages 一致（使用相同的校验规则），支持所有类型的消息（role=[`system`, `user`, `tool`, `assistant`]，支持 `name`、`tool_call_id`、`tool_calls` 参数，不支持 `partial` 参数）。此外，当你的 messages 包含 role 为 `tool` 的消息时，请确保在 role=`tool` 的消息前正确放置了携带 `tool_calls` 参数的 assistant message，并确保该 assistant message 中的所有 `tool_calls` 均被正确放置在 messages 中，否则会导致缓存创建失败。
tools	List[Dict[str, any]]	否	缓存的工具内容，其格式与 `/v1/chat/completions` 接口中的 tools 一致（使用相同的校验规则）。工具列表可以为空（此时你需要保证 `messages` 字段为合法值）。此外，当你的 messages 包含携带 `tool_calls` 参数的 assistant message 时，请确保该 `tool_calls` 中的所有 tools 均已由 `tools` 参数正确提供，否则会导致缓存创建失败。
name	str	否	缓存名称，这是一个辅助性质的字段，可以使用与你业务相关的信息来设置本次缓存的名称。
description	str	否	缓存描述信息，这是一个辅助性质的字段，可以使用与你业务相关的信息来设置本次缓存的描述信息，在检索缓存时，你可以通过 `description` 字段来判断这个缓存是否是你需要的缓存。
metadata	List[Dict[str, str]]	否	缓存的元信息，你可以将与你业务相关的各种信息以 key-value 的形式存储在 `metadata` 中，你最多可以设置 16 组元信息，每组元信息

参数名称	参数类型（以 Python Type Hint 为例）	是否必填	参数说明
			的 key 长度最高不超过 64 个 utf-8 字符，每组元信息的 value 长度最高不超过 512 个 utf-8 字符。
expired_at	int	是（`expired_at` 与 `ttl` 参数必须指定其中的一个值）	缓存的过期时间，格式为 unix 时间戳（单位为秒），是指缓存过期的某个具体时间点（而不是时间段）。注意，请确保 `expired_at` 字段的值大于服务器接收到创建缓存请求时当前时间戳的值，否则会导致缓存创建失败。推荐的做法是，使用当前时间戳加上你希望缓存存活的时间（秒为单位）作为 `expired_at` 字段的值。**额外的，如果不设置 `expired_at` 或该值为 0，我们会为缓存设置一个默认的过期时间，当前为 24 小时。`expired_at` 字段的值最大不超过服务器接收到创建缓存请求的时间点加 3600 秒。**当使用 `expired_at` 参数时，请勿指定 `ttl` 参数。
ttl	int		缓存的有效期，单位为秒，指的是从当前服务器接收到请求的时间点开始，该缓存的存活时间。当使用 `ttl` 参数时，请勿指定 `expired_at` 参数。其与 `expired_at` 的关系为：`expired_at = now() + ttl`

注：当前版本，对于单个用户，创建的单个缓存大小最大为 128k，如果请求中的 `messages` 和 `tools` 字段包含的 Tokens 数量超过 128k，将会导致缓存创建失败。

以下为一个正确的请求示例：

```
{
```

```
"model": "moonshot-v1",
"messages": [
    {
        "role": "system",
        "content": "你是 Kimi，由 Moonshot AI 提供的人工智能助手，你更擅长中文和英文的对话。你会为用户提供安全，有帮助，准确的回答。
    },
    { "role": "user", "content": "你好，我叫李雷，1+1等于多少？" }
],
"tools": [
    {
      "type": "function",
      "function": {
        "name": "CodeRunner",
        "description": "代码执行器，支持运行 python 和 javascript 代码",
        "parameters": {
          "properties": {
            "language": {
              "type": "string",
              "enum": ["python", "javascript"]
            },
            "code": {
              "type": "string",
              "description": "代码写在这里"
            }
          },
          "type": "object"
        }
      }
    }
],
"name": "The name of the cache. Optional. The maximum length is 256 characters",
"description": "The description of the assistant. Optional. The maximum length is 512 characters.",
"metadata": {
  "biz_id": "110998541001"
},
```

```
    "expired_at": 1718680442
}
```

对于上述请求，`/v1/caching` 接口将返回：

```
{
    "id": "cache-id-xxxxxxxxxxxxx",
    "status": "pending",
    "object": "context-cache",
    "created_at": 1699063291,
    "tokens": 32768,
    "expired_at": 1718680442,
    "model": "moonshot-v1",
    "messages": [
        {
            "role": "system",
            "content": "你是 Kimi，由 Moonshot AI 提供的人工智能助手，你更擅长中文和英文的对话。你会为用户提供安全，有帮助，准确的回答。
        },
        { "role": "user", "content": "你好，我叫李雷，1+1等于多少？" }
    ],
    "tools": [
        {
          "type": "function",
          "function": {
            "name": "CodeRunner",
            "description": "代码执行器，支持运行 python 和 javascript 代码",
            "parameters": {
              "properties": {
                "language": {
                  "type": "string",
                  "enum": ["python", "javascript"]
                },
```

```
                "code": {
                  "type": "string",
                  "description": "代码写在这里"
                }
              },
              "type": "object"
            }
          }
        }
    ],
    "name": "The name of the cache. Optional. The maximum length is 256 characters",
    "description": "The description of the assistant. Optional. The maximum length is 512 characters.",
    "metadata": {
      "biz_id": "110998541001"
    }
}
```

返回参数

注：返回值中的`model`、`messages`、`tools`、`name`、`description`、`metadata`参数与创建缓存时的请求参数相同，故在此省略。

参数名称	参数类型（以 Python Type Hint 为例）	参数说明
id	str	缓存 `id`，使用这个 `id` 执行对缓存的 Modify、Retrieve 操作，或是在 `/v1/chat/completions` 接口中携带这个 `id` 以应用缓存。

参数名称	参数类型（以 Python Type Hint 为例）	参数说明
status	Literal["pending", "ready", "error", "inactive"]	当前缓存的状态，其遵循以下规则：1. 当缓存被初次创建时，其初始状态为 `pending`；2. 如果参数合法，缓存创建成功，其状态变更为 `ready`；3. 如果参数不合法，或因其他原因缓存创建失败，其状态变更为 `error`；4. 对于已过期的缓存，其状态变更为 `inactive`；5. 更新一个存在 `id` 的缓存，其状态会重新回到 `pending`，并应用上述步骤 2、3、4；
object	str	当前缓存的存储类型，为固定值 `context_cache`
created_at	int	当前缓存的创建时间
expired_at	int	当前缓存的过期时间
tokens	int	当前已缓存的 Tokens 数量。注意，已缓存的 Tokens 数量并不总是等于最终在 `/v1/chat/completions` 接口中消耗的 Tokens 数量，这是因为在调用 `/v1/chat/completions` 接口时会使用不同的模型（这会影响 Tokens 计算），最终的 Tokens 数以 `/v1/chat/completions` 接口返回的 Usages 信息为准。
error	Dict[str, str]	当缓存创建失败，即 `status` 字段为“error”时，会额外携带一个 `error` 字段，用于表示缓存创建失败的具体失败原因。其具体格式为：`{ "type": "error_type", "message": "error_message"}`

列举 Cache

```
GET https://api.moonshot.cn/v1/caching?limit=20&order=desc&after=cache-id-xxxxxx&metadata[biz_id]=110998541001
```

请求参数

注：请求参数以 URL 查询参数的形式提供

参数名称	参数类型（以 Python Type Hint 为例）	是否必填	参数说明
limit	int	否	指当前请求单页返回的缓存数量，默认值为 20
order	Literal["asc", "desc"]	否	指当前请求时查询缓存的排序规则，按缓存的 `created_at` 进行排序，默认值为 `desc`。
after	str	否	指当前请求时，应该从哪一个缓存开始进行查找，其值为缓存 `id`；**注意，查询结果不包含 `after` 指定的那个缓存。**
before	str	否	指当前请求时，应该查询到哪一个缓存为止，其值为缓存 `id`；**注意，查询结果不包含 `before` 指定的那个缓存。**
metadata	Dict[str, str]	否	指当前请求时，用于筛选缓存的 `metadata` 信息，你可以使用 `metadata` 来快速使用你的业务信息筛选需要的缓存。**使用形如 `metadata[key]=value` 的形式来传递参数。**

以下为一次正确请求返回的内容：

```
{
    "object": "list",
    "data": [
        {
            "id": "cache-id-xxxxxxxxxxxxx",
```

```
"status": "pending",
"object": "context-cache",
"created_at": 1699063291,
"tokens": 32768,
"expired_at": 1718680442,
"model": "moonshot-v1",
"messages": [
    {
        "role": "system",
        "content": "你是 Kimi，由 Moonshot AI 提供的人工智能助手，你更擅长中文和英文的对话。你会为用户提供安全，有帮助，准
    },
    { "role": "user", "content": "你好，我叫李雷，1+1等于多少？" }
],
"tools": [
    {
      "type": "function",
      "function": {
        "name": "CodeRunner",
        "description": "代码执行器，支持运行 python 和 javascript 代码",
        "parameters": {
          "properties": {
            "language": {
              "type": "string",
              "enum": ["python", "javascript"]
            },
            "code": {
              "type": "string",
              "description": "代码写在这里"
            }
          },
          "type": "object"
        }
      }
    }
],
```

```
            "name": "The name of the cache. Optional. The maximum length is 256 characters",
            "description": "The description of the assistant. Optional. The maximum length is 512 characters.",
            "metadata": {
              "biz_id": "110998541001"
            }
        }
    ]
}
```

删除 Cache

```
DELETE https://api.moonshot.cn/v1/caching/{{cache-id}}
```

删除缓存：

1. 如果成功删除已有缓存，则会返回 HTTP 200，并产生一个形如这样的响应信息：

```
{
    "deleted": true,
    "id": "cache-xxxxxxxxxxxxxxxxxxxx",
    "object": "context_cache_object.deleted"
}
```

2. 如果指定的缓存 `id` 不存在，则会返回 `HTTP 404 Not Found`；

更新 Cache

```
PUT https://api.moonshot.cn/v1/caching/{{cache-id}}
```

请求参数

参数名称	参数类型（以 Python Type Hint 为例）	是否必填	参数说明
metadata	List[Dict[str, str]]	否	缓存的元信息，你可以将与你业务相关的各种信息以 key-value 的形式存储在 `metadata` 中，你最多可以设置 16 组元信息，每组元信息的 key 长度最高不超过 64 个 utf-8 字符，每组元信息的 value 长度最高不超过 512 个 utf-8 字符。
expired_at	int	否（`expired_at` 与 `ttl` 参数最多指定其中的一个值）	缓存的过期时间，格式为 unix 时间戳（单位为秒），是指缓存过期的某个具体时间点（而不是时间段）。注意，请确保 `expired_at` 字段的值大于服务器接收到创建缓存请求时当前时间戳的值，否则会导致缓存创建失败。推荐的做法是，使用当前时间戳加上你希望缓存存活的时间（秒为单位）作为 `expired_at` 字段的值。**额外的，如果不设置 `expired_at` 或该值为 0，我们会为缓存设置一个默认的过期时间，当前为 24 小时。`expired_at` 字段的值最大不超过服务器接收到创建缓存请求的时间点加 3600 秒。**当使用 `expired_at` 参数时，请勿指定 `ttl` 参数。

参数名称	参数类型（以 Python Type Hint 为例）	是否必填	参数说明
ttl	int	否（`expired_at` 与 `ttl` 参数最多指定其中的一个值）	缓存的有效期，单位为秒，指的是从当前服务器接收到请求的时间点开始，该缓存的存活时间。当使用 `ttl` 参数时，请勿指定 `expired_at` 参数。其与 `expired_at` 的关系为：`expired_at = now() + ttl`

以下为一个正确的请求示例：

```
{
  "metadata": {
    "biz_id": "110998541001"
  },
  "expired_at": 1718680442
}
```

成功调用更新缓存接口将会返回与创建缓存接口相同的响应内容。

查询 Cache

```
GET https://api.moonshot.cn/v1/caching/{{cache-id}}
```

查询缓存：

1. 缓存 `id` 存在的场合，返回该 `id` 对应的缓存信息，其内容与创建缓存接口返回的响应一致；
2. 若缓存 `id` 不存在，则返回 `HTTP 404 Not Found`；

验证并使用 Cache

缓存将会被应用在 `/v1/chat/completions` 接口中。

👊 关于使用缓存的重要提示

当你在调用 `/v1/chat/completions` 接口时指定了一个合法的尚未过期的缓存 `id` 的场合，我们**不保证**这个缓存 `id` 对应的缓存一定会被启用，在某些特殊场合会出现无法使用缓存的情况，在这种情况下，请求仍然会成功，内容也会正确返回；但当缓存未被启用时，当前请求的 Tokens 及对应的费用均会以 `/v1/chat/completions` 接口的标准资费信息进行计算和扣减。

通过 Headers 使用缓存

为了最大限度地减少对接口、SDK 兼容性的破坏，我们将会通过 HTTP Headers 来验证、使用缓存、以及调整缓存相关规则。

⚠ 注意事项

通过 Headers 调用缓存的场合，你必须：

1. 保证调用 `/v1/chat/completions` 时的 messages 的前 N 个 messages 与缓存的所有 messages 完全一致（N = 缓存的 messages 长度），包括 messages 的顺序，message 中的字段值都需要保持一致，否则会导致无法命中缓存；
2. 保证调用 `/v1/chat/completions` 时的 tools 与缓存的 tools 完全一致，否则将会导致无法命中缓存；

注 1：我们会使用 Hash 来校验请求 `/v1/chat/completions` 接口的 messages 前缀与缓存的 messages 是否一致、以及请求的 tools 与缓存的 tools 是否一致，因此请务必保证这两者与缓存内容保持完全一致。

注 2：在通过 Headers 使用缓存的场合，即使消息已经被缓存，你也必须在请求 `/v1/chat/completions` 时再次携带这些消息。

与缓存相关的请求 Headers

Headers 名称	是否必填	Headers 说明
X-Msh-Context-Cache	是	通过设置该 Header 来指定当前请求所使用的缓存，其值为缓存 `id`，**只有设置了该值才会启用缓存。**
X-Msh-Context-Cache-DryRun	否	通过设置该 Header 来指定仅验证缓存是否生效，而不执行推理过程，其值为 `Literal[1, 0]`；**如果值为 1，则只验证缓存是否生效，而不执行推理过程，也不消耗任何 Tokens。**
X-Msh-Context-Cache-Reset-TTL	否	通过设置该值来自动延长缓存的 `expired_at` 过期时间，其值为单位为秒的时间段整数；如果设置了该值，则每次成功调用 `/v1/chat/completions` 接口都会为该缓存设置新的有效期，新的有效期时间为服务器接收到请求的时间点加上该 Header 指定的值。**例如，当 TTL 设置为 86400 时，每次请求成功，都会将缓存的过期时间置为 `now() + 86400`，而非 `expired_at + 86400`。**额外的，如果当前缓存已过期，在设置了该 Header 的场合，会重新启用该缓存，并将其状态设置为 `pending` 并更新其 `expired_at`。

与缓存相关的响应 Headers

Headers 名称	Headers 说明
Msh-Context-Cache-Id	当前请求所使用的缓存 `id`
Msh-Context-Cache-Token-Saved	当前请求由于使用了缓存所节省的 Tokens 数量
Msh-Context-Cache-Token-Exp	当前缓存的过期时间，即 `expired_at`

以下是一个使用缓存调用 `/v1/chat/completions` 的正确示例：

```
POST https://api.moonshot.cn/v1/chat/completions

Content-Type: application/json; charset=utf-8
Content-Length: 6418
X-Msh-Context-Cache: cache-id-xxxxxxxxxxxxxxx
X-Msh-Context-Cache-Reset-TTL: 86400

{
    "model": "moonshot-v1-128k",
    "messages": [
        {
            "role": "system",
            "content": "你是 Kimi，由 Moonshot AI 提供的人工智能助手，你更擅长中文和英文的对话。你会为用户提供安全，有帮助，准确的回答。
        },
        { "role": "user", "content": "你好，我叫李雷，1+1等于多少？" }
    ],
    "tools": [
        {
          "type": "function",
```

```
            "function": {
              "name": "CodeRunner",
              "description": "代码执行器，支持运行 python 和 javascript 代码",
              "parameters": {
                "properties": {
                  "language": {
                    "type": "string",
                    "enum": ["python", "javascript"]
                  },
                  "code": {
                    "type": "string",
                    "description": "代码写在这里"
                  }
                },
                "type": "object"
              }
            }
        }
    ]
}
```

Python 用户使用

```
from openai import OpenAI

client = OpenAI(
    api_key = "$MOONSHOT_API_KEY",
    base_url = "https://api.moonshot.cn/v1",
)

completion = client.chat.completions.create(
```

```
    model = "moonshot-v1-8k",
    messages = [
        {"role": "system", "content": "你是 Kimi，由 Moonshot AI 提供的人工智能助手，你更擅长中文和英文的对话。你会为用户提供安全，有
        {"role": "user", "content": "你好，我叫李雷，1+1等于多少？"}
    ],
    temperature = 0.3,
    extra_headers={
        "X-Msh-Context-Cache": "cache-xxx-xxx-xxx",
        "X-Msh-Context-Cache-Reset-TTL": "86400",
    },
)

print(completion.choices[0].message.content)
```

💊 通过 Message Content 使用缓存

你可以将形如以下形式的 message 置于你的 messages 列表的首位以使用缓存：

```
{
    "role": "cache",
    "content": "cache_id=xxx;other-options"
}
```

这是一个特殊的 `role=cache` 的消息，它通过 `content` 字段指定需要使用哪个缓存：

1. 你必须把这个消息放在 messages 列表的第一位；
2. `tools` 参数必须为 `null` （空数组也将视为有值）；

3. 我们会将这条特殊的消息替换为你已经缓存的消息列表，并将缓存中的 `tools` 填充到 tools 参数中；
4. 你不需要在 messages 中放置已经被缓存的消息，你只需要添加那些没有被缓存的 messages 即可；
5. 你可以将这条特殊的消息视作**引用已缓存的消息列表**；
6. 对于 `content`，其规则如下：
7. 使用 `cache_id={id}` 的形式来指定缓存 `id`；
8. 通过 `reset_ttl={ttl}` 的形式来指定是否重新设置 `expired_at` 过期时间；
9. 通过 `dry_run=1` 的形式来指定是否仅校验缓存而不启用推理过程；
10. 以上参数通过分号 `;` 进行拼接，最后一个参数后请不要再放置一个额外的分号；
11. 其中：

- `cache_id` 对应 Headers 中的 `X-Msh-Context-Cache`；
- `dry_run` 对应 Headers 中的 `X-Msh-Context-Cache-DryRun`；
- `reset_ttl` 对应 Headers 中的 `X-Msh-Context-Cache-Reset-TTL`；
- 参数值及规则也与 Headers 保持一致；

Last updated on June 26, 2024